Exhibit A

Expectation Management Design

Developed By: Jeff Doyel

Overview

This document describes the high-level approach taken to solving the problems related to when an expectation applies to a person, and when an expectation is considered due or over-due for the person.

Definitions

Expectation (EXP): Something that a health practitioner is expected to do with regard to a person being clinically evaluated. The expectation may be any number of things including, but not limited to, ordering a procedure, reviewing a health risk, educating the patient, or more.

Qualification Facts: This is a small subset of information that will be used in the evaluation of the majority of rules. Right now we believe this subset with consists of the person's *gender*, *age*, *problems, and diagnoses*. In the future we will also expand this set to include *clinical facts* (as developed by the knowledge team).

Satisfaction: A record of the fulfillment and completion of one or more action that satisfy a given expectation for a specific person. The possible actions include a broad range of types and events within the system.

Start Up

1. The service will create a cache for Qualification Facts during startup. This cache will be keyed by the PeronId and will include a structure that contains gender, birthdate, problems, and diagnosis.
2. The service will read the following startup parameters and save them in static memory. These parameters will be used to determine when specific parts of the Qualification Facts must be reloaded from the database.
    a. GenderCacheExpireInterval - Interval in seconds
    b. BirthdateCacheExpireInterval - Interval in seconds
    c. ProblemCacheExpireInterval - Interval in seconds
    d. DiagnosisCacheExpireInterval - Interval in seconds
3. The service will load all the reference data for the expectations. It will then compare this data with the current state of the data cached it in global memory. Any discrepancies in the data will be updated with the newest version of the data. Using this technique the cache will be automatically updated whenever any instance of the server is restarted.
4. The service will start a worker thread used to retrieve and manage Qualification Facts.
5. The service will start a worker thread used to…

Service Requests

1. Get Qualified Expectations

The input message will include

PersonId	Double	Required
Gender	Integer	Optional
Birthdate	Date	Optional
Problems[]	List	
NomenclatureId	Double	Optional
Diagnosis[]	List	
NomenclatureId	Double	Optional
Facts[]	List	
NomenclatureId	Double	Future Use

The output message will be a list of EXP that the person qualifies for

Expectation Management Design

Developed By: Jeff Doyel

Expectations[]	List	
ExpectationId	Double	Required

a. If the optional parameters (Current Date/Time and Qualification Facts) are not supplied in the input message the service will retrieve the data.

> The Qualification Facts will be cached in memory for a predefined period of time. Once they expire the cache will be released. The expiration interval will be unique for each of the following Qualification Facts: Gender, Birth Date, Problems, and Diagnoses. This expiration interval will be supplied as a startup property of the service.
>
> The logic used to retrieve the Qualification Facts will be implemented in a separate thread of execution. This Qualification Fact retrieval thread will run continuously waiting for requests to service. Once it finishes servicing a request it will put the Qualification Facts in the cache and will notify the requestor of the availability of the data.

b. Call Discern Expert passing it the full set of Qualification Facts including the gender, age (calculated from birth date), problems, and diagnoses. Base upon the result returned by Discern Expert we would either drop the EXP or keep it.

> It is assumed that a unique qualification rule will exist for each EXP. Since Discern Expert is designed to trigger the execution of all rules that are associated with a given message (request), we can easily evaluate each qualification rule by sending a single message (request) to the Discern Expert server. This will be a new message (request) that will be designed solely for the purpose of evaluating the qualification of health expectations. Discern Expert will need to be reconfigured to recognize and accept this new message.
>
> This new input (request) message will include the following:

PersonId	Double	Required
Gender	Integer	Required
Age	Double	Required
Problems[]	List	
NomenclatureId	Double	Required
Diagnosis[]	List	
NomenclatureId	Double	Required
Facts[]	List	
NomenclatureId	Double	Future Use

> Each qualification rule will be responsible for appending its respective EXP to the output (reply) message with an indicator that shows whether the person qualifies or not.
>
> The new output (reply) message will include the following:

Expectations[]	List	
ExpectationId	Double	Required
Qualifies	Boolean	Required

> The Health Expectation Service will parse the reply to determine which EXPs to return as qualified.

2. Get State of Expectations

> The input message will include:

PersonId	Double	Required
Expectations[]	List	
ExpectationId	Double	Required
EvaluationDateTime	Date/Time	Optional

> The output message will include the same EXPs as the input message with the addition of a status code and next due date for each.

Expectations[]	List	
ExpectationId	Double	Required

StatusCd	Double	Required
NextDue	Date/Time	Optional

A predefined list of orderables, results, and manually satisfied EXPs will be identified as data elements that impact the Satisfaction History.

For each EXP the following steps will need to occur in order to determine the complete history of prior satisfactions.

a. Load the Satisfaction History from Orders

For each SynonymId that is associated with the EXP, the persons orders will need to be evaluated. This will be accomplished by reading the orders from the orders table using a key of SynonymId and PersonId.

For each order found the current_start_dt_tm will be evaluated and compared with other orders found for this EXP. The most recent current_start_dt_tm will be considered the date/time at which this EXP was LAST satisfied through the placement of an order.

b. Load the Satisfaction History from Results

For each ClinicalEventCd that is associated with the EXP, the persons results will need to be evaluated. This will be accomplished by reading the results from the clinical event table using a key of ClinicalEventCd and PersonId.

For each clinical event found the current_start_dt_tm will be evaluated and compared with other clinical events found for this EXP. The most recent current_start_dt_tm will be considered the date/time at which this EXP was LAST satisfied through the receipt of a clinical result.

For each EXP the following steps will need to occur in order to determine the current state and next due date of the EXP.

c. Evaluate each EXP by using the CDC algorithm. This evaluation will identify when the EXP is due again and whether it is currently overdue.

The detailed specifications for the CDC algorithm can be found at http://www.cdc.gov/nip/registry/peg.pdf. Refer to the Expectation Evaluation Algorithm section below for a complete description of the logic.

This algorithm requires the person's satisfaction history as input along with the current date/time.

This algorithm also uses a set of reference data for each EXP being evaluated. Refer to the Expectation Satisfaction Parameter section below for a complete description of this reference data.

The EvaluationDateTime will be used if it is passed in the input message. If it is not included in the input message then the current date/time will be used.

The satisfaction history will include ONLY the clinical results as determine by the logic above.

The CDC algorithm will be called for each EXP. The result of this algorithm will be a status code (Complete, Up_To_Date, No_Recommended, Contraindicated, Wrong_Alternate, or Recommended). The result may also include the next due date/time if applicable.

The output message with be updated to include the status and the next due date/time if applicable.

3. Get Expectation Reminders

The input message will include:

PersonId	Double	Required
Gender	Integer	Optional
Birthdate	Date	Optional
Problems[]	List	
NomenclatureId	Double	Optional

Expectation Management Design

Developed By: Jeff Doyel

Diagnosis[]	List	
NomenclatureId	Double	Optional
Facts[]	List	
NomenclatureId	Double	Future Use

> The output message will be a list of EXPs that are not currently satisfied. For each EXP the status code and next due date will be included.

Expectations[]	List	
ExpectationId	Double	Required
StatusCd	Double	Required
NextDue	Date/Time	Required

a. Internally call the same logic as the "Get Qualified Expectations" request to determine what expectations this person qualifies for.

> This will simply forward the optional parameters on to the "Get Qualified Expectations" request.
>
> The result of this call will be a list of EXPs. These EXPs will be passed into another internal call to the "Get State of Expectations" request.

b. Internally call the same logic as the "Get State of Expectations" request.

> This will simply forward the optional date/time parameter and the list of qualifying EXPs on to the "Get State of Expectations" request.
>
> The result will be a list of EXPs with their corresponding status and next due date/time.

> For each EXP the following steps will need to occur in order to reduce the list of EXP down to the ones that are currently due or overdue.

c. Drop all EXP that are already satisfied.

> Evaluate the status of the EXP. If the status indicates that the EXP is already satisfied then do not include it in the output message.

d. Drop all EXP that are not yet due.

> Evaluate the status of the EXP. If the status indicates that the EXP is not yet due then do not include it in the output message.

e. Include all remaining EXPs.

> All remaining EXPs must be due or over due. Add each one to the output message along with the due date and status.

Service Requests Uses

1. Health Maintenance View.

   The "Get Expectation Reminders" request will be leveraged to get the expectations that are currently due or over-due.

   > The person's gender and birth date will be included in the input message. If the person's problems and/or diagnoses have already been loaded then we will include them in the input message. The current date/time will be included in the input message.

2. Expectation Overdue Reports/Alerts.

   The "Get State of Expectations" request will be leveraged to determine whether a given person is due or over-due for a given EXP.

   > For each situation where a list of persons who are due or overdue for a given EXP is necessary this request will need to be called multiple times (once for each person).

Expectation Management Design

Developed By: Jeff Doyel

> It is assumed that the list of persons who need to be evaluated is known. Often times this list will be a subset of the person population (e.g. Dr. Smith's patients, Clinic A's patients).
>
> We are assuming that the evaluation will NEVER occur against the entire person population.
>
> The logic necessary to retrieve the subset of persons and repeatedly call the request is simply and can easily be accomplished in CCL.

Expectation Satisfaction Parameters

The following parameters have been identified from the CDC Algorithm as necessary attributes for accurate evaluation. Many of these parameters may be unnecessary for some EXPs and as such they are not all required for each EXP.

- Expectation Schedule

Name	String	The name of the schedule
Age of First Step	Double	The …

- Expectation Series

Name	String	The name of the series
Age of First Step	Double	The …
Number of Steps Required	Integer	The number of steps required to complete this series.
Unlimited Steps	Boolean	Indicates that the series include an unlimited number of steps.

- Expectation

Name	String	The name of the expectation
Interval Only	Boolean	Indicates that …
Sequence Number	Integer	The sequence at which the expectations will be recommended when more then one expectation is recommended.
Minimum Age	Double	The minimum age at which this expectation is recommended.
Maximum Age	Double	The maximum age at which this expectation is recommended.

- Expectation Step

Step Number	Integer	The number of the step.
Min. Interval to Count	Long	The interval (in days) between when the previous expectation step was satisfied and when this expectation step can count.
Min. Interval to Administer	Long	The interval (in days) between when the previous expectation step was satisfied and when this expectation step can be safely administered.
Recommended Interval	Long	The interval (in days) between when the previous expectation step was satisfied and when this expectation should be recommended.
Minimum Age	Double	The minimum age at which this expectation step can be satisfied.
Recommended Age	Double	The age at which this expectation step is recommended.
Valid From Age	Double	The minimum age at which this expectation step should be recommended.
Valid To Age	Double	The maximum age at which this expectation step should be recommended.
Skip Age	Double	The age…

Expectation Evaluation Algorithm

The logic of this algorithm was obtained from the Center for Disease Control and Prevention National Immunization Program. This algorithm, although specifically developed for automating

the immunization evaluation process, can also be applied equally well to general health maintenance scenarios. It turns out that this algorithm more then satisfies the basic needs of most health maintenance guidelines.

A detailed explanation of the algorithm logic and the recommended implementation approach can be found at http://www.cdc.gov/nip/registry/peg.pdf.